



Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

AGH University of Krakow

Numerical Methods in Physics

Project 06

Fitting Tabulated Data Using Singular Value Decomposition

Hamzeh Khanpour

hamzeh.khanpour@fis.agh.edu.pl

Faculty of Physics and Applied Computer Science
AGH University of Krakow

Why this laboratory?

Two questions

How do we fit tabulated data with a polynomial using SVD in C++?

How do we verify the matrix reconstruction and assess the fitted curve?

Scientific workflow

Generate data $\longrightarrow$ construct A $\longrightarrow$ compute SVD
$\longrightarrow$ calculate coefficients $\longrightarrow$ verify $\longrightarrow$ interpret.

Course sources

S01: Tomasz Chwiej, Numerical Methods in Physics course page.
S02: Laboratory handout: Fitting tabulated data using SVD.
S03: Lecture: Approximation of function (least-squares fitting).

Technical sources

Netlib LAPACKE: C interface to LAPACK (row-major matrix support).
LAPACK routine: `DGESVD` (singular value decomposition of a real matrix).
Implementation: `LAPACKE_dgesvd()` with thin SVD.



Two questions we want to answer

How do we determine the coefficients of a polynomial from tabulated data?

How does singular value decomposition help us solve that problem and check the result?

$$\text{Data} \longrightarrow \text{matrix } A \longrightarrow \text{SVD} \longrightarrow \text{coefficients}$$
$$\longrightarrow \text{fitted curve} \longrightarrow \text{numerical verification}$$

Division of responsibilities

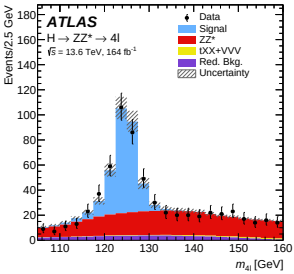
We teach the mathematics using five complete *small examples*. We implement the official $x \cos x$ assignment independently in the laboratory.

www.agh.edu.pl

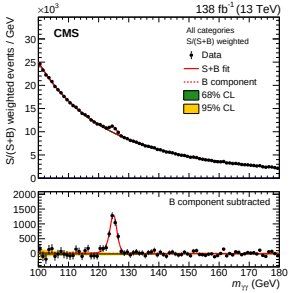
A real example of fitting: Higgs signals at the LHC

We measure an invariant mass for each selected collision event. The **shape of the distribution** helps distinguish Higgs signal from background.

ATLAS (2026): $H \to ZZ^* \to 4\ell$
$13.6\,\mathrm{TeV}$; $164\,\mathrm{fb}^{-1}$



CMS (2026): $H \to \gamma\gamma$
$13\,\mathrm{TeV}$; $138\,\mathrm{fb}^{-1}$



Why this matters for numerical methods

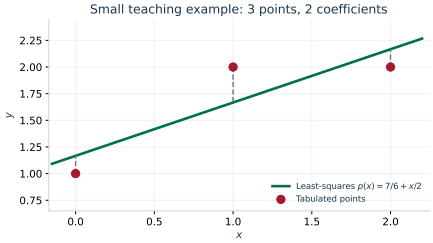
The peak near $125\,\mathrm{GeV}$ is the Higgs signal. ATLAS compares data with pre-fit predictions; CMS fits signal plus background. Particle-physics fits use likelihoods, while our lab uses SVD least squares.

Sources: ATLAS HIGP-2025-09 Fig. 4; CMS HIG-24-007 Fig. 6b

Why do physicists fit data?

Measurements are individual pairs (x_i, y_i). A model summarizes a trend and lets us compare measurements with theory.

- calibrate detectors and sensors
- estimate parameters from noisy measurements
- interpolate or predict cautiously between measurements



Physical interpretation

A fitted curve is an approximation: we must still examine **residuals**, model choice and numerical sensitivity.

Interpolation

Seek a function that passes through **every** listed data point, if the points and model allow it.

Least-squares fitting

Choose a model with fewer adjustable parameters than data points and find coefficients that make the overall discrepancy small.

For our laboratory: $m > n$, $\quad A\mathbf{c} \approx \mathbf{y}$.

With m measurements and n unknown coefficients, an exact solution is generally unavailable. A least-squares solution is the appropriate target.

Learning map: from mathematics to code



1. **E01**: build a tiny polynomial design matrix.
2. **E02**: calculate fitted values, residuals and a least-squares check.
3. **E03**: rebuild a known 2×2 matrix using an exact SVD.
4. **E04**: call `LAPACKE_dgesvd()` on an independent 4×2 matrix.
5. **E05**: fit three simple points using SVD.

Optional E06

Study how an unusually small singular value amplifies noise. The complete assessed polynomial-fitting program remains our own task.

Notation

A data table contains m pairs (x_i, y_i) for $i = 0, \ldots, m-1$.

$$\mathbf{x} = (x_0, \ldots, x_{m-1})^T, \qquad \mathbf{y} = (y_0, \ldots, y_{m-1})^T.$$

Example: three observations

point i	0	1	2
x_i	0	1	2
y_i	1	2	2

x_i tells us where we sample the system; y_i is the value to be approximated.



A model with linear unknown coefficients

$$\boxed{p(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1}}$$

- n is the number of unknown coefficients.
- $n - 1$ is the polynomial degree.
- The model can be nonlinear in x and still be **linear in the coefficients** c_j.

Two parameters

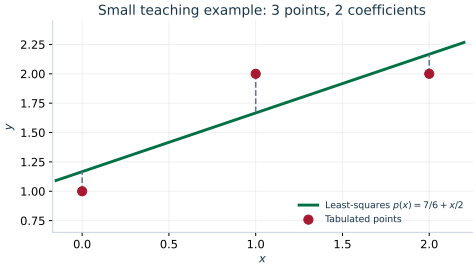
$n = 2$ gives a straight line: $p(x) = c_0 + c_1 x$.
We will learn SVD using this simpler model before the official polynomial.

$$(0,1), \quad (1,2), \quad (2,2)$$

Our model is $p(x) = c_0 + c_1 x$.

Three measurements; two unknown coefficients.

Question: which straight line gives the best overall agreement?



What E01 teaches

How the powers $1, x_i, x_i^2, \ldots$ become columns of a matrix. No SVD library is needed yet.

For $p(x) = c_0 + c_1 x$, insert each observation coordinate:

$$\begin{aligned} x_0 = 0: &\quad c_0 + 0c_1 \approx 1, \\ x_1 = 1: &\quad c_0 + 1c_1 \approx 2, \\ x_2 = 2: &\quad c_0 + 2c_1 \approx 2. \end{aligned}$$

Stack the left sides as $A\mathbf{c}$:

$$\underbrace{\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix}}_{A\ (3\times 2)} \underbrace{\begin{pmatrix} c_0 \\ c_1 \end{pmatrix}}_{\mathbf{c}\ (2\times 1)} \approx \underbrace{\begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}}_{\mathbf{y}\ (3\times 1)}.$$

AGH

The general rule

$$\boxed{A_{ij} = x_i^j, \qquad 0 \le i < m, \quad 0 \le j < n.}$$

Each **row** belongs to a data point; each **column** belongs to a polynomial power.

E01: check by hand

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix}.$$

The first column is always $x_i^0 = 1$ (including $x_i = 0$, as a constant basis function).

C++17 row-major array: `a[i*n+j]`. E01 prints every entry for verification.

www.agh.edu.pl

Why not solve all three equations exactly?

If $c_0 = 1$ from the first point and $c_0 + c_1 = 2$ from the second, then $c_1 = 1$; the third predicts $1 + 2(1) = 3$, not 2.

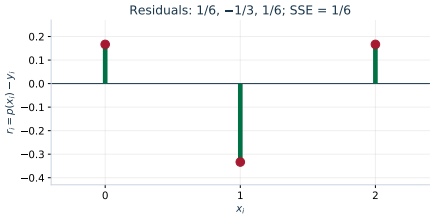
$$\underbrace{\text{3 observations}}_{m=3} > \underbrace{\text{2 parameters}}_{n=2}.$$

Important distinction

We do *not* force $A\mathbf{c} = \mathbf{y}$. We choose coefficients that minimize the collective error, $\|A\mathbf{c} - \mathbf{y}\|_2^2$.

Residuals: measure the mismatch

$$r_i = p(x_i) - y_i, \qquad \mathbf{r} = A\mathbf{c} - \mathbf{y}.$$



For the least-squares line $p(x) = \frac{7}{6} + \frac{1}{2}x$:

$$\mathbf{r} = \begin{pmatrix} 1/6 \\ -1/3 \\ 1/6 \end{pmatrix}.$$

Positive r_i: fitted curve is above the point. Negative r_i: it is below.

E02 prints y_i, $p(x_i)$ and each residual side by side.

Add squared residuals, not signed residuals

$$S(c_0, c_1) = \sum_{i=0}^{m-1} \left[p(x_i) - y_i\right]^2 = \|A\mathbf{c} - \mathbf{y}\|_2^2.$$

For our tiny data set:

$$S = (c_0 - 1)^2 + (c_0 + c_1 - 2)^2 + (c_0 + 2c_1 - 2)^2.$$

Why square the differences?

Positive and negative residuals cannot cancel, and the function is smooth enough to optimize mathematically.

Differentiate S with respect to c_0 and c_1, and set both derivatives to zero:

$$\begin{pmatrix} 3 & 3 \\ 3 & 5 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \end{pmatrix}.$$

$$\boxed{c_0 = \frac{7}{6}, \quad c_1 = \frac{1}{2}, \quad S_{\min} = \frac{1}{6}.}$$

Verify rather than merely trust

$\mathbf{r} = (1/6, -1/3, 1/6)^T$ and $A^T\mathbf{r} = (0,0)^T$. E02 independently checks both identities.

The normal equations are used *only* to obtain an exact reference for this toy problem; the official project uses SVD.

AGH

A numerical difficulty

The normal equations are $A^T A\mathbf{c} = A^T\mathbf{y}$. If A has full column rank, its 2-norm condition number obeys

$$\kappa_2(A^T A) = \kappa_2(A)^2.$$

When A is already poorly conditioned, squaring its condition number can severely amplify floating-point errors.

Our strategy

Compute the SVD of A itself, use its singular vectors and singular values, and check both reconstruction and fit quality.

AGH

The central factorization

$$\boxed{A = U\Sigma V^T.}$$

- V^T: right-singular-vector coordinates of the input.
- Σ: scales each independent singular direction.
- U: expresses the result along left singular directions.

Intuition

An arbitrary linear transformation can be described as an orthogonal change of directions, independent stretches, and another orthogonal change of directions.

Thin SVD: always check the dimensions

For our $m \times n$ design matrix with $m > n$:

$$\underbrace{A}_{m\times n} = \underbrace{U}_{m\times n}\,\underbrace{\Sigma}_{n\times n}\,\underbrace{V^T}_{n\times n}.$$

object	shape, for $m = 4, n = 2$	interpretation
A	4×2	measurement-to-parameter map
U	4×2	2 orthonormal columns
Σ	2×2	singular values on diagonal
V^T	2×2	right singular vectors in rows

Check: $(4 \times 2)(2 \times 2)(2 \times 2)$ yields (4×2).

The columns of U: left singular vectors

$$U = (\mathbf{u}_0\ \mathbf{u}_1\ \cdots\ \mathbf{u}_{n-1}), \qquad \mathbf{u}_k \in \mathbb{R}^m.$$

Orthonormality

$$\mathbf{u}_k^T \mathbf{u}_l = \delta_{kl}, \qquad U^T U = I_n.$$

For a thin SVD with $m > n$, UU^T is a projector onto the column space of A, *not* generally the full $m \times m$ identity.

Why is U important for fitting?

The scalar $\mathbf{u}_k^T \mathbf{y}$ measures how much of the data vector points along singular direction k.

$$V = (\mathbf{v}_0 \ \cdots \ \mathbf{v}_{n-1}), \qquad V^T V = I_n.$$

Coefficient-space directions

Each $\mathbf{v}_k \in \mathbb{R}^n$ describes one independent combination of the unknown polynomial coefficients.

An essential LAPACKE detail

The output array is called `vt`: it contains V^T. Each right singular vector $\mathbf{v}_k^T$ is a **row** of this array.

$$(\mathbf{v}_k)_j = (V^T)_{kj}.$$

Singular values: scaling strengths

$$\Sigma = \mathrm{diag}(\sigma_0, \ldots, \sigma_{n-1}), \qquad \boxed{\sigma_0 \geq \sigma_1 \geq \cdots \geq \sigma_{n-1} \geq 0.}$$

- A large σ_k represents a well-resolved direction.
- A very small σ_k means a small data change may cause a large coefficient change.
- A zero singular value indicates rank deficiency.

Numerical conditioning

For full-column-rank A: $\kappa_2(A) = \sigma_0/\sigma_{n-1}$.

AGH

Vector length and projection

$$\|\mathbf{a}\|_2 = \sqrt{\sum_i a_i^2}, \qquad \mathbf{a}^T\mathbf{b} = \sum_i a_i b_i.$$

Unit vectors $\mathbf{q}_1, \mathbf{q}_2$ are orthogonal when $\mathbf{q}_1^T\mathbf{q}_2 = 0$.

Two-dimensional illustration

$$\mathbf{q}_1 = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}, \qquad \mathbf{q}_2 = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}, \qquad \mathbf{q}_1^T\mathbf{q}_2 = 0.$$

This dot-product operation is exactly what E05 uses to project data onto the columns of U.

Take an independent, easy-to-check matrix:

$$A = \begin{pmatrix} 0 & 3 \\ 1 & 0 \end{pmatrix}.$$

One valid SVD is

$$\boxed{U = I_2, \quad \Sigma = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}, \quad V^T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.}$$

Two singular values

$\sigma_0 = 3$, $\sigma_1 = 1$. Check the exact matrix product on the next slide.

$$U\Sigma V^T = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \boxed{\begin{pmatrix} 0 & 3 \\ 1 & 0 \end{pmatrix}}.$$

A test we can actually compute

$$\varepsilon_{\mathrm{rec}} = \|A - U\Sigma V^T\|_F = 0 \quad \text{(exact arithmetic)}.$$

E03 uses these *known* factors. It is not a general-purpose SVD solver; E04 introduces LAPACKE.

AGH

Multiply a column by a row

$$\begin{pmatrix} a \\ b \end{pmatrix} \begin{pmatrix} c & d \end{pmatrix} = \begin{pmatrix} ac & ad \\ bc & bd \end{pmatrix}.$$

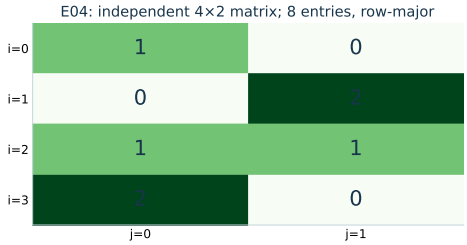
The result has rank at most one (unless it is the zero matrix).

Reconstruction as singular contributions

$$\boxed{A = \sum_{k=0}^{n-1} \sigma_k \mathbf{u}_k \mathbf{v}_k^T.}$$

Each term rebuilds one independent part of the original matrix.

Thin SVD for a rectangular matrix



E04 uses

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 1 \\ 2 & 0 \end{pmatrix}.$$

Thin dimensions:

$$U : 4 \times 2,\ \Sigma : 2 \times 2,\ V^T : 2 \times 2.$$

Why this example?

We practice genuine LAPACKE calls on a separate 4×2 matrix, before tackling the assessed polynomial matrix.

$$A_{ij} = \sum_{k=0}^{n-1} \sigma_k \, U_{ik} (V^T)_{kj}.$$

For a 4×2 matrix, each entry uses two terms

$$A_{i0} = \sigma_0 U_{i0} (V^T)_{00} + \sigma_1 U_{i1} (V^T)_{10}.$$

C++: the same indices

```
reconstructed[i*n+j] += s[k] * u[i*n+k] * vt[k*n+j];
```

Compare the reconstructed elements with a saved, unmodified copy of A.

How can we verify a numerical SVD?

1. **Reconstruction:** $\|A - U\Sigma V^T\|_F$ should be small relative to $\|A\|_F$.
2. **Left orthogonality:** $\|U^T U - I_n\|_F$ should be small.
3. **Right orthogonality:** $\|VV^T - I_n\|_F$ should be small.
4. **Ordering:** $\sigma_0 \geq \cdots \geq \sigma_{n-1} \geq 0$.

Expected ambiguity

Singular vectors may change sign together (and degenerate subspaces may rotate). Compare reconstruction and invariants, **not** arbitrary vector signs.

A dot product creates one number

$$\alpha_k = \mathbf{u}_k^T \mathbf{y} = \sum_{i=0}^{m-1} U_{ik} y_i.$$

Illustration

For $\mathbf{u} = (3/5, 4/5)^T$ and $\mathbf{y} = (5, 0)^T$,

$$\mathbf{u}^T \mathbf{y} = (3/5)5 + (4/5)0 = 3.$$

Projection asks: how strongly is the observed data aligned with the k-th independent singular direction?

$$
\begin{aligned}
A &= U\Sigma V^T, \\
A^+ &= V\Sigma^{-1}U^T \qquad \text{(full column rank)}, \\
\mathbf{c} &= A^+\mathbf{y} = V\Sigma^{-1}U^T\mathbf{y}.
\end{aligned}
$$

Why it minimizes the residual

$AA^+ = UU^T$ projects $\mathbf{y}$ onto the column space of A. The remaining residual is orthogonal to every column of A.

If a singular value vanishes, use Σ^+ (reciprocals only for retained nonzero singular values) rather than a nonexistent inverse.

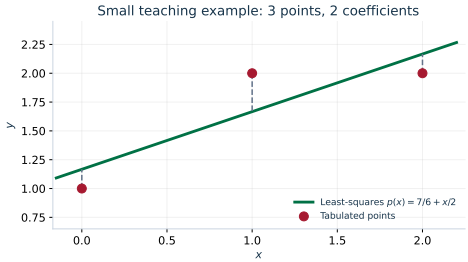
The coefficient formula as a sum

$$\mathbf{c} = \sum_{k=0}^{n-1} \frac{\mathbf{u}_k^T \mathbf{y}}{\sigma_k} \mathbf{v}_k \qquad \text{(full column rank).}$$

1. Compute the projection $\alpha_k = \mathbf{u}_k^T \mathbf{y}$.
2. Divide by the corresponding singular value σ_k.
3. Multiply by right singular vector $\mathbf{v}_k$.
4. Add the vectors to form the polynomial coefficients.

For zero / numerical-rank-deficient singular values use the pseudoinverse convention; do not divide blindly.

E05: a complete toy fit using SVD



$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}.$$

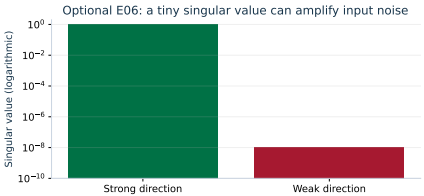
Use thin SVD and obtain

$$\boxed{\mathbf{c} = \begin{pmatrix} 7/6 \\ 1/2 \end{pmatrix}}.$$

Independent numerical test

E05 runs LAPACKE, evaluates the singular-vector sum, and checks $S = 1/6$ and $A^T\mathbf{r} \approx 0$.

When a singular value is very small



The coefficient contribution contains $1/\sigma_k$. If $\sigma_k = 10^{-8}$, a data perturbation of order 10^{-6} along that direction may affect coefficients by $\sim 10^{2}$.

Optional E06

A small independent 2×2 example shows the difference between retaining and discarding a weak singular direction. Truncation changes the answer; it is not a magic fix.

Two independent checks

SVD reconstruction: $\varepsilon_A = \|A - U\Sigma V^T\|_F / \|A\|_F$.

Fit residual: $\varepsilon_y = \|A\mathbf{c} - \mathbf{y}\|_2$.

Interpretation

A tiny ε_A verifies that the decomposition reconstructs the *matrix*. A small ε_y shows agreement with the *sampled data*; neither guarantees good behavior between samples.

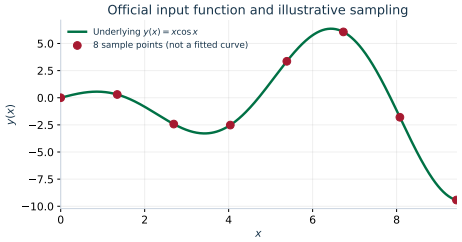
These are helpful numerical diagnostics; the official instructions explicitly require reconstruction and the fitting comparison plots.



- A degree-$n-1$ polynomial has n adjustable coefficients.
- Increasing n enlarges the fitting space, but does not guarantee better behavior between points.
- Monomials $1, x, x^2, \ldots$ can have widely different magnitudes over $0 \leq x \leq 3\pi$.
- This may cause a large condition number and sensitivity to floating-point rounding.

Good scientific practice

Inspect **both** residuals at the measured nodes and a densely evaluated curve on the full interval.



$$y(x) = x\cos x, \quad 0 \le x \le 3\pi.$$

The graph shows the known function and an illustration of eight equispaced nodes. **No assessed polynomial fit is shown.**

Important concept

$y(x)$ is nonlinear in x, but $p(x) = \sum_j c_j x^j$ is linear in the *unknown* coefficients c_j. This is a linear least-squares problem, not Newton root finding.

AGH

Row-major linear indexing

$$\boxed{A_{ij} \longleftrightarrow \texttt{a[i*n+j]}.}$$

For $m = 3, n = 2$, rows $(1,0), (1,1), (1,2)$ become

```
a = {1,0, 1,1, 1,2}.
```

What to check

The outer loop advances through data points i; the inner loop advances through polynomial powers j. The stride between rows is the number of columns n.

Introducing LAPACKE_dgesvd()



The LAPACKE entry point

```
lapack_int LAPACKE_dgesvd(
   int matrix_layout, char jobu, char jobvt,
   lapack_int m, lapack_int n, double* a, lapack_int lda,
   double* s, double* u, lapack_int ldu,
   double* vt, lapack_int ldvt, double* superb);
```

Our choices for $m \geq n$

`LAPACK_ROW_MAJOR`, `jobu='S'`, `jobvt='S'`: store thin singular-vector factors and singular values.

Read the singular vectors correctly

$$(\mathbf{u}_k)_i = \texttt{u[i*n+k]}, \qquad (\mathbf{v}_k)_j = \texttt{vt[k*n+j]}.$$

object	array size	leading dimension
A	mn	`lda=n`
U	mn	`ldu=n`
V^T	nn	`ldvt=n`

Very common mistake

`vt` stores V^T, not V. Accessing `vt[j*n+k]` instead of `vt[k*n+j]` generally produces the wrong coefficient formula.



- Preserve the original matrix: LAPACKE may overwrite its input array.
- Allocate mn entries for A and U; n^2 for V^T; n singular values.
- Allocate a `superb` buffer safely (standard minimum $\max(1, n-1)$ for this case).
- Check `info`: zero means success; nonzero must be investigated.
- Use `std::vector<double>` and avoid compiler-specific variable-length arrays.

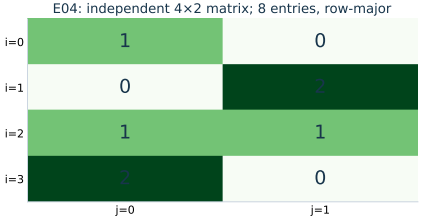
E04 and E05

Compile with a C++17 compiler, `lapacke.h`, and appropriate LAPACKE/LAPACK/BLAS link libraries.

www.agh.edu.pl

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 1 \\ 2 & 0 \end{pmatrix}, \quad m = 4,\ n = 2.$$

$$A = U_{4\times 2}\Sigma_{2\times 2}V^T_{2\times 2}.$$



What E04 checks

Descending singular values, $U^T U \simeq I$, $VV^T \simeq I$, and $U\Sigma V^T \simeq A$.

No polynomial fitting of the official $x \cos x$ data is implemented here.

AGH

Loop-level mathematical steps

For every retained singular direction k:

$$\alpha_k = \sum_{i=0}^{m-1} \texttt{u[i*n+k]}\; y_i, \qquad c_j \mathrel{+}= \frac{\alpha_k}{s_k}\, \texttt{vt[k*n+j]}.$$

Small, verifiable numerical target

E05 recovers $c_0 = 7/6$ and $c_1 = 1/2$ from three points. Cross-check $S = 1/6$; test programs return a failure code when checks do not pass.

A numerical-rank tolerance prevents division by exactly zero or numerically negligible singular values.

AGH

Required initial parameters

$$\boxed{m = 8, \quad n = 6, \quad x_a = 0, \quad x_b = 3\pi.}$$

$$x_i = x_a + i\frac{x_b - x_a}{m - 1}, \qquad y_i = x_i \cos x_i.$$

Matrix dimensions

A has 8 rows and 6 columns; **y** has 8 entries; **c** has 6 entries; the fitted polynomial is degree 5.

Our task is to implement these steps independently using the methods demonstrated in E01–E05.

Official workflow: SVD and coefficients



1. Build and save $A_{ij} = x_i^j$ and preserve its original contents.
2. Calculate thin SVD of A with `LAPACKE_dgesvd()`.
3. Save all computed singular values to file.
4. Reconstruct and check $A_{ij} = \sum_k \sigma_k U_{ik} (V^T)_{kj}$.
5. Calculate and save $\mathbf{c} = \sum_k (\mathbf{u}_k^T \mathbf{y}/\sigma_k)\mathbf{v}_k$ (with appropriate zero checks).

The assessed programming exercise

These steps are requirements, not a supplied complete source file. Integrate the learned methods in your **own** C++ program.

run	data points m	coefficients n	polynomial degree
laboratory	8	6	5
homework A	30	10	9
homework B	30	20	19

Three separate figures in our report

Show input points and the fitted polynomial for each pair (m, n). A dense x grid makes behavior *between* the points visible.

Explain differences between input data, curve shape, residuals and conditioning. The official sample figure with $(30, 15)$ is illustrative only, *not* a required homework run.


THANK
YOU!